University of Pittsburgh

Algorithms and Data Structures 2
CS 1501

Spring 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming deadlines:
  - Homework 5 due on 2/21
  - Lab 5 due on 2/25
  - Homework 6 due on 2/28
  - Assignment 1 due on 3/14
- Midterm exam on Wednesday 3/2
  - In-person, paper, closed book exam

Previous lecture …

- DLB Trie
- Lossless Compression
  - Huffman Compression

CourseMIRROR Reflections (most interesting)

- The most interesting thing was the examples and exercises. They help a lot
- I found the use of Linked lists in DLBs as a nice way to save space compared to Multi Branch RSTs
- DLB tries were interesting and I liked their relation to linked lists
- Nodelets are interesting! Looking forward to learning more about compression
- the run time and space comparisons of all of the trees we've learned about
- how dlb trees can store so much information
- The different ways we can store the data in nodes along trees or tries
- RSTs with linked lists
- The different kinds of tries was most interesting.

CourseMIRROR Reflections (most confusing)

- The lecture went very fast today.
- How are nodelets converted into DLB nodes?
- Could you go over how Huffman Codes work? The expansion at the end felt rushed
- why we use a trie over the other trees. Or when we should use one.
- Compression was went over quickly so Im not sure I grasped the basics yet
- implementation of the multi-way RST
- Id like to have a general overview of the different trees/tries and a code walkthrough

Problem of the Day: Lossless Compression

- Input: A sequence of characters
  - *n* characters
  - each encoded as an 8-bit Extended ASCII
- Output: A bit string
  - of length less than 8*n
  - the original sequence can be fully restored from the bitstring

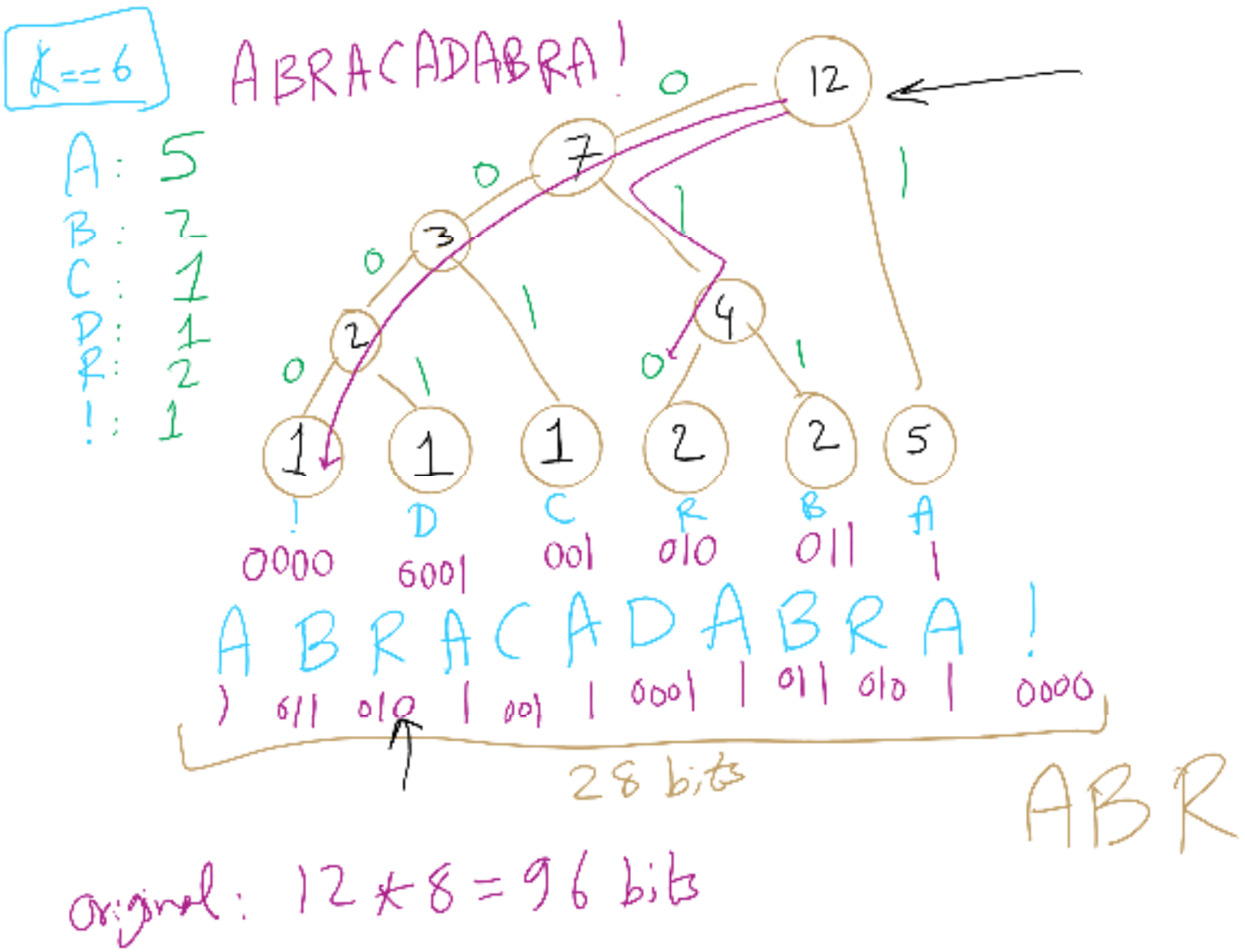
Subproblem: Prefix-free Compression

- Input: A sequence of n characters
- Output: A codeword h_i for each character i such that
  - No codeword is a prefix of any other
  - When each character in the input sequence is replaced with each codeword
    - the length of that compressed sequence is minimum
    - the original sequence can be fully restored from the compressed bitstring

Generating Huffman codes

- Assume we have K characters that are used in the file to be compressed and each has a weight (its frequency of use)
- Create a forest, F, of K single-node trees, one for each character, with the single node storing that char's weight
- while |F| > 1:
  - Select T1, T2 ∈ F that have the smallest weights in F
  - Create a new tree node N whose weight is the sum of T1 and T2's weights
  - Add T1 and T2 as children (subtrees) of N
  - Remove T1 and T2 from F
  - Add the new tree rooted by N to F
- Build a tree for "ABRACADABRA!"

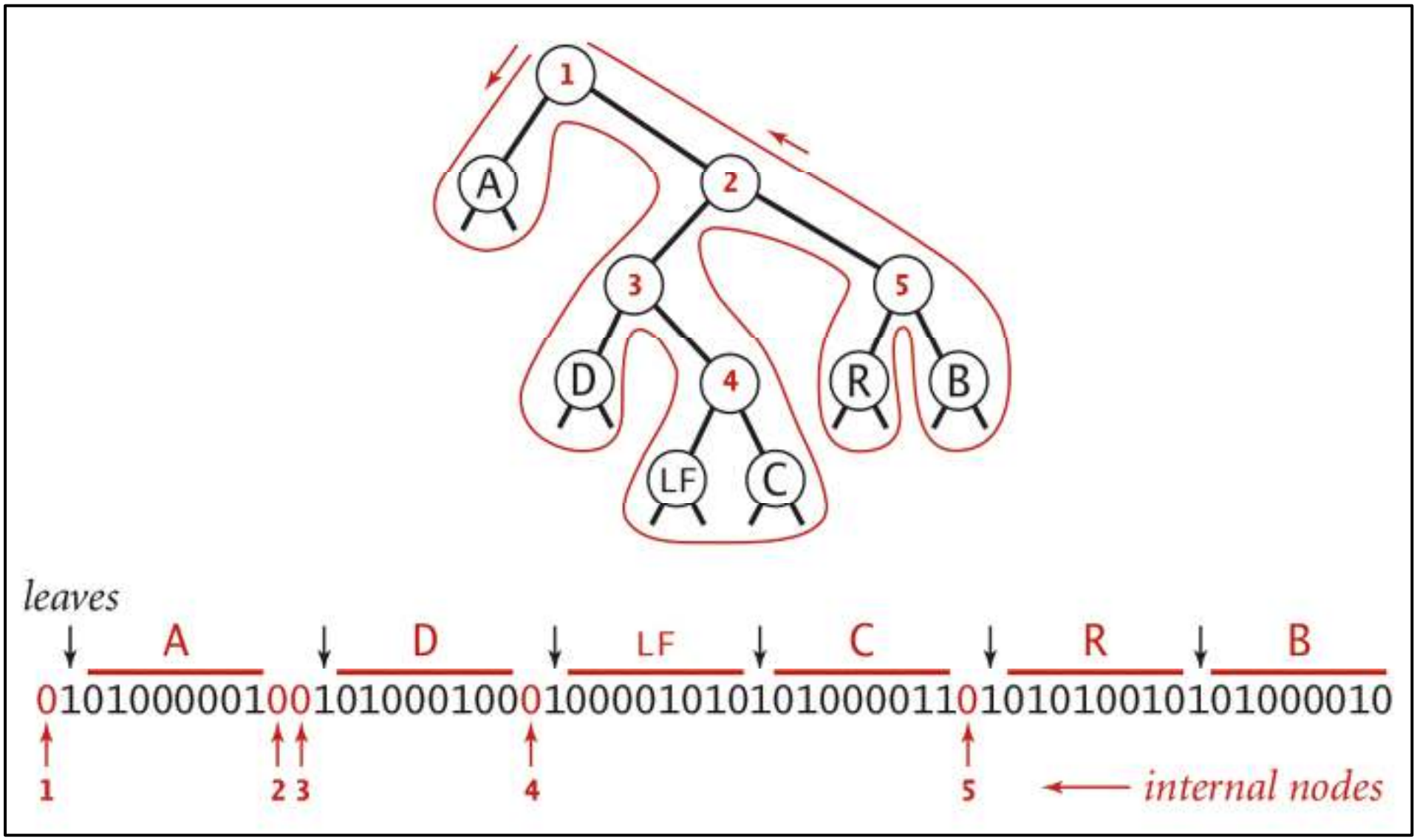
Huffman Tree Construction Example



Implementation concerns

- To encode/decode, we'll need to read in characters and output codes/read in codes and output characters
  - …
  - Sounds like we'll need a symbol table!
    - What implementation would be best?
      - Same for encoding and decoding?
  - Note that this means we need access to the trie to expand a compressed file!

Representing tries as bitstrings



Binary I/O

```
private static void writeTrie(Node x){
    if (x.isLeaf()) {
        BinaryStdOut.write(true);
        BinaryStdOut.write(x.ch);
        return;
    }
    BinaryStdOut.write(false);
    writeTrie(x.left);
    writeTrie(x.right);
}

private static Node readTrie() {
    if (BinaryStdIn.readBoolean())
       return new Node(BinaryStdIn.readChar(), 0, null, null);
    return new Node('\0', 0, readTrie(), readTrie());
}
```

Binary I/O

```
private static void writeBit(boolean bit) {
      // add bit to buffer
      buffer <<= 1;
      if (bit) buffer |= 1;
      // if buffer is full (8 bits), write out as a single byte
      N++;
      if (N == 8) clearBuffer();
}
```

```
writeBit(true);
writeBit(false);
writeBit(true);
writeBit(false);
writeBit(false);
writeBit(false);
writeBit(false);
writeBit(true);
```

buffer: **00000000**

N: **0**

Please submit your reflections by using the CourseMIRROR App

If you are having a problem with CourseMIRROR, please send an email to coursemirror.development@gmail.com

8/29/2022